IN THE CLAIMS:

Please amend the claims such that the pending claims read as follows:

1. (Amended) A method of managing a file system for a file server, comprising the steps of:

receiving a file operation that signals a reservation operation for a file having a file size;

computing a number of blocks needed to be reserved to accommodate the file; and

reserving for later allocation a number of unallocated blocks in the file system equal to the number of blocks needed to be reserved to accommodate the file.

2. A method as in claim 1, wherein the file system uses a write anywhere file system layout.

3. A method as in claim 1, wherein the file operation that signals the reservation operation is a zero length write request.

4. A method as in claim 1, wherein the file operation that signals the reservation operation includes a parameter that specifies the file size.

5. A method as in claim 1, wherein the step of computing the number of blocks needed to be reserved to accommodate the file further comprises:

determining a total number of direct and indirect blocks needed to accommodate the file size; and

subtracting a total number of blocks already allocated for the file and a total number of cached unallocated blocks for the file from the total number of direct and indirect blocks needed to accommodate the file size.

6. (Amended) A method as in claim 1, wherein the step of reserving for later allocation the number of unallocated blocks in the file system equal to the number of blocks needed further comprises:

setting a flag in an inode for the file that indicates blocks have been reserved for the file; and

incrementing a reserved block count in a file system information block by the number of blocks needed, the reserved block count indicating how many unallocated blocks have been reserved for files in the file system.

7. (Amended) A method as in claim 1, further comprising the step of checking that a number of available blocks in the file system is greater than the number of blocks needed to be reserved to accommodate the file, wherein an error is returned in a case that the number of available blocks is less than the number of blocks needed.

8. A method as in claim 7, wherein the number of available blocks in the file system is determined by subtracting a number of allocated blocks, a number of cached unallocated blocks, and a number of reserved blocks from a total number of blocks in the file system, and adding a number of reserved cached unallocated blocks.

9. (Amended) A method as in claim 1, further comprising the step of checking that the number of blocks needed to be reserved to accommodate the file does not exceed a remainder of a quota for an owner of the file, wherein an error is returned in a case that the number of blocks needed exceeds the remainder of the quota.

10. A method as in claim 1, further comprising the step of releasing reservation of blocks as those blocks are written to storage.

11. A method as in claim 10, wherein the step of releasing reservation of blocks further comprises the step of decrementing a reserved block count in a file system information block by a number of released blocks, the reserved block count indicating how many unallocated blocks have been reserved for files in the file server.

SUB B1> 12. (Amended) A method of managing a file system for a file server, comprising the steps of:

receiving a file operation that signals a reservation operation for a file for which reservation has already been performed, said reservation operation specifying a new file size different from a current file size for the file;

comparing the current file size with the new file size;

in the case that the current file size exceeds the new file size, releasing any remaining block reservations for the file; and

in the case that the new file size exceeds the current file size, reserving for later allocation in the file system an additional number of unallocated blocks equal to a difference between a total number of direct and indirect blocks required by the new file size and a total number of direct and indirect blocks required by the current file size.

13. A method as in claim 12, wherein the file system uses a write anywhere file system layout.

14. A method as in claim 12, wherein the file operation that signals the reservation operation is a zero length write request.

15. A method as in claim 12, wherein the file operation that signals the reservation operation includes a parameter that specifies the file size.

16. A method as in claim 12, wherein the step of releasing remaining block reservations for the file further comprises the steps of:

resetting a flag in an inode for the file that indicates blocks have been reserved for the file; and

decrementing a reserved block count in a file system information block by a number of blocks still reserved for the file, the reserved block count indicating how many unallocated blocks have been reserved for files in the file system.

17. A method as in claim 12, further comprising the step of checking that a number of available blocks in the file system is greater than the additional number of unallocated blocks, wherein an error is returned in a case that the number of available blocks is less than the additional number of blocks.

18. A method as in claim 17, wherein the number of available blocks in the file system is determined by subtracting a number of allocated blocks, a number of cached unallocated blocks, and a number of reserved blocks from a total number of blocks in the file system, and adding a number of reserved cached unallocated blocks.

19. (Amended) A method as in claim 12, further comprising the step of checking that the additional number of blocks does not exceed a remainder of a quota for an owner of the

file, wherein an error is returned in a case that the additional number of blocks exceeds the remainder of the quota.

20. A method as in claim 12, further comprising the step of releasing reservation of blocks as those blocks are written to storage.

21. A method as in claim 20, wherein the step of releasing reservation of blocks further comprises the step of decrementing a reserved block count in a file system information block by a number of released blocks, the reserved block count indicating how many blocks total have been reserved for files in the file server.

22. (New) A file server that manages a file system, comprising:

mass storage that stores files for the file system;

a processor that executes instructions to manage the file system; and

a memory storing information including instructions, the instructions executable by the processor to manage the file system, the instructions comprising: (a) receiving a file operation that signals a reservation operation for a file having a file size; (b) computing a number of blocks needed to be reserved to accommodate the file; and (c) reserving for later allocation a number of unallocated blocks in the file system equal to the number of blocks needed to be reserved to accommodate the file.

23. (New) A file server as in claim 22, wherein the file system uses a write anywhere file system layout.

24. (New) A file server as in claim 22, wherein the file operation that signals the reservation operation is a zero length write request.

25. (New) A file server as in claim 22, wherein the file operation that signals the reservation operation includes a parameter that specifies the file size.

26. (New) A file server as in claim 22, wherein the step of computing the number of blocks needed to be reserved to accommodate the file further comprises:

determining a total number of direct and indirect blocks needed to accommodate the file size; and

subtracting a total number of blocks already allocated for the file and a total number of cached unallocated blocks for the file from the total number of direct and indirect blocks needed to accommodate the file size.

27. (New) A file server as in claim 22, wherein the step of reserving for later allocation the number of unallocated blocks in the file system equal to the number of blocks needed further comprises:

setting a flag in an inode for the file that indicates blocks have been reserved for the file; and

incrementing a reserved block count in a file system information block by the number of blocks needed, the reserved block count indicating how many unallocated blocks have been reserved for files in the file system.

28. (New) A file server as in claim 22, wherein the instructions further comprise the step of checking that a number of available blocks in the file system is greater than the number of blocks needed to be reserved to accommodate the file, and wherein an error is returned in a case that the number of available blocks is less than the number of blocks needed.

29. (New) A file server as in claim 28, wherein the number of available blocks in the file system is determined by subtracting a number of allocated blocks, a number of cached unallocated blocks, and a number of reserved blocks from a total number of blocks in the file system, and adding a number of reserved cached unallocated blocks.

30. (New) A file server as in claim 22, wherein the instructions further comprise the step of checking that the number of blocks needed to be reserved to accommodate the file does not exceed a remainder of a quota for an owner of the file, and wherein an error is returned in a case that the number of blocks needed exceeds the remainder of the quota.

31. (New) A file server as in claim 22, wherein the instructions further comprise the step of releasing reservation of blocks as those blocks are written to storage.

32. (New) A file server as in claim 31, wherein the step of releasing reservation of blocks further comprises the step of decrementing a reserved block count in a file system information block by a number of released blocks, the reserved block count indicating how many unallocated blocks have been reserved for files in the file server.

SUB B2

33. (New) A file server that manages a file system, comprising:

mass storage that stores files for the file system;

a processor that executes instructions to manage the file system; and

a memory storing information including instructions, the instructions executable by the processor to manage the file system, the instructions comprising: (a) receiving a file operation that signals a reservation operation for a file for which reservation has already been performed, said reservation operation specifying a new file size different from a current file size for the file; (b) comparing the current file size with the new file size; (c) in the case that the current file size exceeds the new file size, releasing any remaining block reservations for the file; and (d) in the case that the new file size exceeds the current file size, reserving for later allocation in the file system an additional number of unallocated blocks equal to a difference between a total number of direct and indirect blocks required by the new file size and a total number of direct and indirect blocks required by the current file size.

34. (New) A file server as in claim 33, wherein the file system uses a write anywhere file system layout.

35. (New) A file server as in claim 33, wherein the file operation that signals the reservation operation is a zero length write request.

36. (New) A file server as in claim 33, wherein the file operation that signals the reservation operation includes a parameter that specifies the file size.

37. (New) A file server as in claim 33, wherein the step of releasing remaining block reservations for the file further comprises the steps of:

resetting a flag in an inode for the file that indicates blocks have been reserved for the file; and

decrementing a reserved block count in a file system information block by a number of blocks still reserved for the file, the reserved block count indicating how many unallocated blocks have been reserved for files in the file system.

38. (New) A file server as in claim 33, wherein the instructions further comprise the step of checking that a number of available blocks in the file system is greater than the additional number of unallocated blocks, and wherein an error is returned in a case that the number of available blocks is less than the additional number of blocks.

39. (New) A file server as in claim 38, wherein the number of available blocks in the file system is determined by subtracting a number of allocated blocks, a number of cached unallocated blocks, and a number of reserved blocks from a total number of blocks in the file system, and adding a number of reserved cached unallocated blocks.

40. (New) A file server as in claim 33, wherein the instructions further comprise the step of checking that the additional number of blocks does not exceed a remainder of a quota for an owner of the file, and wherein an error is returned in a case that the additional number of blocks exceeds the remainder of the quota.

41. (New) A file server as in claim 33, wherein the instructions further comprise the step of releasing reservation of blocks as those blocks are written to storage.

42. (New) A file server as in claim 41, wherein the step of releasing reservation of blocks further comprises the step of decrementing a reserved block count in a file system information block by a number of released blocks, the reserved block count indicating how many blocks total have been reserved for files in the file server.

43. (New) A memory storing information including instructions, the instructions executable by a processor to manage a file system, the instructions comprising the steps of:

receiving a file operation that signals a reservation operation for a file having a file size;

computing a number of blocks needed to be reserved to accommodate the file; and

reserving for later allocation a number of unallocated blocks in the file system equal to the number of blocks needed to be reserved to accommodate the file.

44. (New) A memory as in claim 43, wherein the file system uses a write anywhere file system layout.

45. (New) A memory as in claim 43, wherein the file operation that signals the reservation operation is a zero length write request.

46. (New) A memory as in claim 43, wherein the file operation that signals the reservation operation includes a parameter that specifies the file size.

47. (New) A memory as in claim 43, wherein the step of computing the number of blocks needed to be reserved to accommodate the file further comprises:

determining a total number of direct and indirect blocks needed to accommodate the file size; and

subtracting a total number of blocks already allocated for the file and a total number of cached unallocated blocks for the file from the total number of direct and indirect blocks needed to accommodate the file size.

48. (New) A memory as in claim 43, wherein the step of reserving for later allocation the number of unallocated blocks in the file system equal to the number of blocks needed further comprises:

setting a flag in an inode for the file that indicates blocks have been reserved for the file; and

incrementing a reserved block count in a file system information block by the number of blocks needed, the reserved block count indicating how many unallocated blocks have been reserved for files in the file system.

49. (New) A memory as in claim 43, wherein the instructions further comprise the step of checking that a number of available blocks in the file system is greater than the number of blocks needed to be reserved to accommodate the file, and wherein an error is returned in a case that the number of available blocks is less than the number of blocks needed.

50. (New) A memory as in claim 49, wherein the number of available blocks in the file system is determined by subtracting a number of allocated blocks, a number of cached

unallocated blocks, and a number of reserved blocks from a total number of blocks in the file system, and adding a number of reserved cached unallocated blocks.

51. (New) A memory as in claim 43, wherein the instructions further comprise the step of checking that the number of blocks needed to be reserved to accommodate the file does not exceed a remainder of a quota for an owner of the file, and wherein an error is returned in a case that the number of blocks needed exceeds the remainder of the quota.

52. (New) A memory as in claim 43, wherein the instructions further comprise the step of releasing reservation of blocks as those blocks are written to storage.

53. (New) A memory as in claim 52, wherein the step of releasing reservation of blocks further comprises the step of decrementing a reserved block count in a file system information block by a number of released blocks, the reserved block count indicating how many unallocated blocks have been reserved for files in the file server.

SUB B37 54. (New) (Amended) A memory storing information including instructions, the instructions executable by a processor to manage a file system, the instructions comprising the steps of:

receiving a file operation that signals a reservation operation for a file for which reservation has already been performed, said reservation operation specifying a new file size different from a current file size for the file;

comparing the current file size with the new file size;

in the case that the current file size exceeds the new file size, releasing any remaining block reservations for the file; and

in the case that the new file size exceeds the current file size, reserving for later allocation in the file system an additional number of unallocated blocks equal to a difference between a total number of direct and indirect blocks required by the new file size and a total number of direct and indirect blocks required by the current file size.

55. (New) A memory as in claim 54, wherein the file system uses a write anywhere file system layout.

56. (New) A memory as in claim 54, wherein the file operation that signals the reservation operation is a zero length write request.

57. (New) A memory as in claim 54, wherein the file operation that signals the reservation operation includes a parameter that specifies the file size.

58. (New) A memory as in claim 54, wherein the step of releasing remaining block reservations for the file further comprises the steps of:

resetting a flag in an inode for the file that indicates blocks have been reserved for the file; and

decrementing a reserved block count in a file system information block by a number of blocks still reserved for the file, the reserved block count indicating how many unallocated blocks have been reserved for files in the file system.

59. (New) A memory as in claim 54, wherein the instructions further comprise the step of checking that a number of available blocks in the file system is greater than the additional number of unallocated blocks, and wherein an error is returned in a case that the number of available blocks is less than the additional number of blocks.

60. (New) A memory as in claim 59, wherein the number of available blocks in the file system is determined by subtracting a number of allocated blocks, a number of cached unallocated blocks, and a number of reserved blocks from a total number of blocks in the file system, and adding a number of reserved cached unallocated blocks.

61. (New) A memory as in claim 54, wherein the instructions further comprise the step of checking that the additional number of blocks does not exceed a remainder of a quota

for an owner of the file, and wherein an error is returned in a case that the additional number of blocks exceeds the remainder of the quota.

62. (New) A memory as in claim 54, wherein the instructions further comprise the step of releasing reservation of blocks as those blocks are written to storage.

63. (New) A memory as in claim 62, wherein the step of releasing reservation of blocks further comprises the step of decrementing a reserved block count in a file system information block by a number of released blocks, the reserved block count indicating how many blocks total have been reserved for files in the file server.